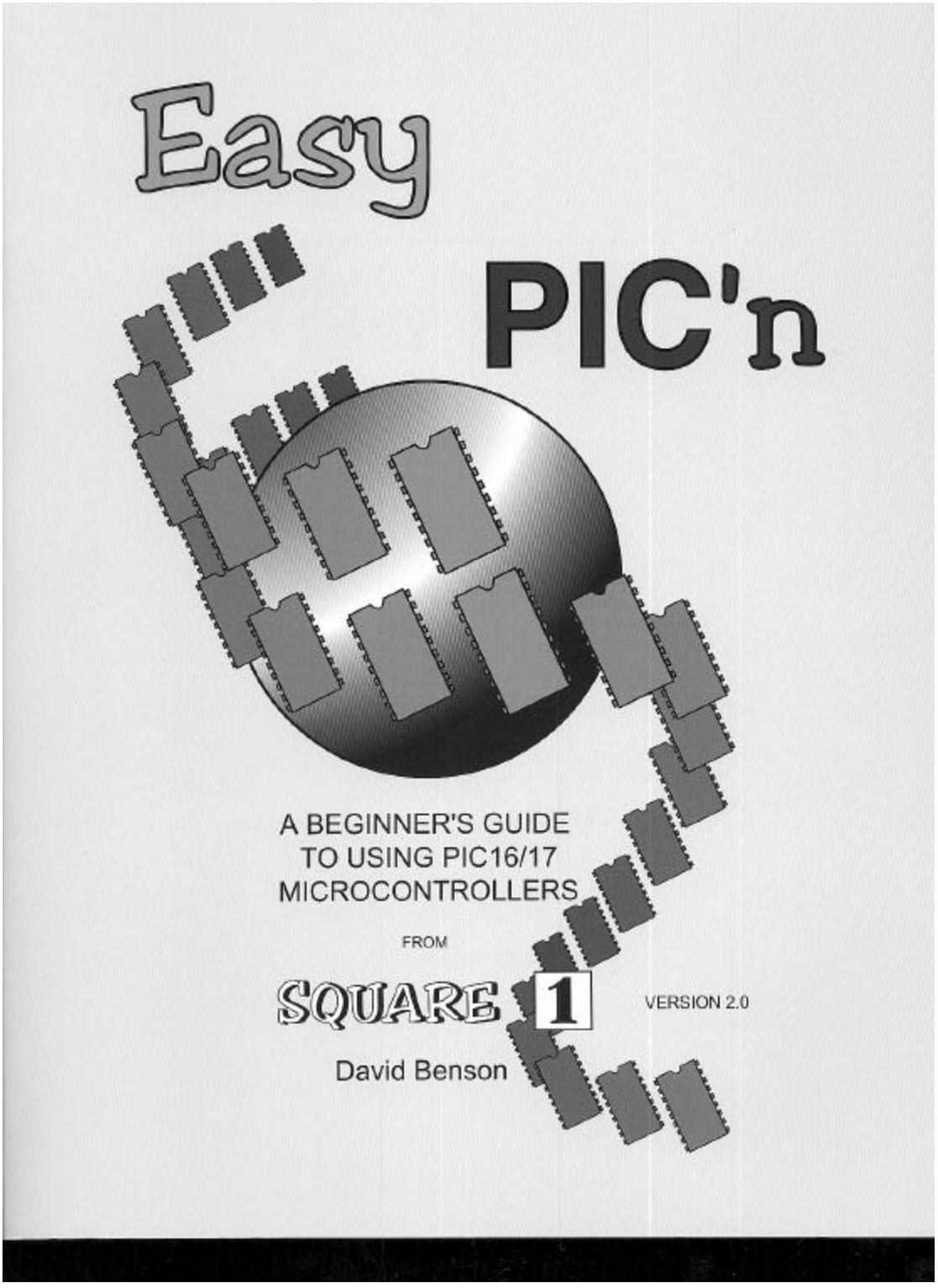
Jameco Part number 136645



Jameco Part number 136645

A BEGINNER'S GUIDE TO USING

PIC16/17 MICROCONTROLLERS

FROM

SQUARE 1

INTRODUCTION 1
PIC16/17 PRODUCT OVERVIEW 3
PIC16C84 4

- Pins and Functions 4
- Package 4
- Clock Oscillator 5
- Configuration "Fuses" 6
- Reset 6
- Ports 7
- Special Features 7
  - Watchdog Timer 7
  - Power-up Timer 7
  - Sleep Mode 7
- Circuit For Experiments 8
- PIC16/17 Architecture 9
  - Program Memory 9
  - Weird Hex Notation 9
  - File Registers 10
  - Working Register (W) 10
  - Option Register 10
  - Stack 10
  - Reset Vector 10
  - Interrupt Vector 10
  - Option Register 12
  - Program Counter 13
  - Status Register 15

ASSEMBLERS 16

- Source Code For The Assembler 16
  - Semicolon (;) 17
  - Tabs 18
- Style 19
- Headers 20
- Labels 20

Equates 21
Literals 22
Origin 23
End 24
Program Format 24
Files Used By Assembler 25
Files Created By Assembler 25
Preventing Some Gotchas 27

FIRING UP THE MPASM ASSEMBLER AND PICSTART 29

How To Assemble A Program 32
How To Use The PICSTART Programmer 35

WRITING PROGRAMS 37

Programming Concepts 37
Instruction Set 40
Instruction Format For The Assembler 43
Byte-Oriented Instructions 43
Bit-Oriented Instructions 44
Literal Instructions 44
Control Instructions (CALL and GOTO) 44
Destination Designator (d) 45
Hexadecimal Numbers vs. MPASM Assembler 46
Binary And Decimal Numbers vs. MPASM Assembler 47
Addressing Modes 48
Immediate addressing 48
Direct Addressing 48
Indirect Addressing 48
Relative Addressing 49
Use Of The RETLW Instruction For Accessing Tables of Data Via Relative Addressing 50
Using The Ports 52
Data Direction 52
Port Read/Write 52
Flags 53
Simple Data Transfers 54
Loop – Endless 55
Loop With a Counter 56
Loop Until 59
Comparisons 60
Bit Manipulation Using Bit Manipulation Instructions 63
Bit Manipulation Using Logic Instructions 65
Using Bit Manipulation 67
Sequencing 71
Subroutines 74
Time Delay Loop 76
Lookup Tables 85

Jameco Part number 136645

INTERRUPTS **88**

Interrupt Control Register 89
Interrupt Sources 90
External 90
Timer/Counter 90
Port B Interrupt On Change - Bits 7,6,5,4 91
Global Interrupt Enable Flag 91
Saving Status During An Interrupt (Context Saving) 91
Where To Put The Interrupt Service Routine 92
Interrupt Latency 93
Multiple Interrupt Sources 94
Example - External Interrupt 95

TIMING AND COUNTING **101**

Digital Output Waveforms 101
Timing And Counting Using The 16C84's TMRO On-Board Timer/Counter 102
Prescaler 104
Changing Prescaler Assignment 105
Using The Timer/Counter 106
Timer/Counter Experiments 107
Digital Output Using TMR0 - Internal Clock 107
Single Time Interval - Internal Clock 109
Single Time Interval - External Clock 112
Free Running Mode - Internal Clock 115
Free Running Mode - External Clock 119
Counting Events (Pulses) 120

PIC16C54 **125**

Pins And Functions 126
Packages 126
Clock Oscillator 127
Configuration "Fuses" 127
Ports 127
Architecture 128
Program Memory 128
File Registers 128
Stack 129
Reset Vector 129
Program Counter 129
Option Register 130
Status Register 131
No Timer/Counter Overflow Interrupt/Flag Output 131
Fewer Instructions 131
PIC16C54 Programming Example 132
Timing And Counting 133
Timer/Counter Description 135
Using The Timer/Counter 136
Timer/Counter Experiment 136

Jameco Part number 136645

MENDING OUR WAYS **137**

Sink vs. Source 137
File Register Bank Switching 138
Bank Switching Demo 139
Interrupts And Bank Switching 142
Program Memory Paging 142
Include Files 142

I/O CONVERSION **145**

Input Conversion 145
Output Conversion 148

APPENDICES

Appendix A – Sources 151
Appendix B – Hexadecimal Numbers 152

Jameco Part number 136645

INTRODUCTION

The PIC16/17 was originally designed as a Peripheral Interface Controller (PIC) for a 16-bit microprocessor. It was essentially an I/O controller and was designed to be very fast. It had a small microcoded instruction set. This design became the basis for Microchip Technology's PIC16C5X family of microcontrollers.

The PIC16C84 microcontroller is unique because its program memory is EEPROM. It can be programmed, tested in-circuit and reprogrammed if necessary in a matter of a few minutes and without the need for a UV EPROM eraser. It is a small device (18-pin), readily available to all including hobbyists and students at a cost of $6.50 (at this writing) in single quantity.

Think of the PIC16C84 as a custom I/O handler. It looks at its inputs and, based on what it sees, it sends signals out its outputs. You can customize it to do what you want via programming. It is not a heavy duty number cruncher or data manipulator.

The MPASM assembler from Microchip Technology (manufacturer of PIC16/17 family) is the PIC16/17 assembler of choice for use in this book because:

- It is free from Microchip
- Most people who are into PIC16/17's speak "MPASM"
- Most examples in magazines and on the Internet are written in the MPASM dialect

A variety of programmers are available for the PIC16C84 (see sources in Appendix A). Some range in cost from $40 to $70. Some come with their own assemblers but will accept hex files created by MPASM (details later). Microchip makes several PIC16/17 programmers at this writing. The PICSTART 16B1 is the low end model which will program base-line PIC16C5X (18 and 28-pin) and 18-pin mid-range PIC16C61, PIC16C62X, PIC16C71, PIC16C84 micro-controllers. The PICSTART 16B1 is used as the example in this book.

The PIC16C84, MPASM assembler and a low cost PIC16C84 programmer make a very inexpensive development system and a great way to get started creating microcontroller based projects.

What you learn using the PIC16C84 is directly applicable to the whole line of PIC16/17 micro-controllers. The approach taken with the PIC16C84 in the first part of book is backwards compatible with the PIC16C54 (explained toward the end of the book). The hassles of file register bank switching and program memory paging are avoided as they are confusing to a beginner but manageable after some experience is gained.

Jameco Part number 136645

Learning how PIC16/17's work and how to apply them involves study in three areas.

- MS DOS computer use (as needed)
- Assembler
- PIC16/17 itself

PIC16/17's are not easy to write code for by hand because they are not intuitive. Many instructions require setting or clearing a bit to specify the destination for the result of executing the instruction. This is ok as an initial learning experience, but would become tedious very soon. So use of "power tools" is essential. This means learning to use an MS-DOS computer if you haven't already done so. It also means learning to use an assembler which converts English-like readable instructions into machine language understood by the PIC16/17 itself.

Finally, learning about the PIC16/17's inner workings is possible once use of the power tools is understood.

The object is to make this process as easy and enjoyable as possible. Once you get through this and you have programmed a PIC16C84 for the first time, a whole new world awaits. You will be able to create more interesting projects and have more fun!

The usual approach to teaching the use of the PIC16/17 is to assume total knowledge of MS-DOS, get into all of the assembler commands and then show advanced examples. There is no simple example showing how to get started. As usual, only 5 percent of this information is needed to get started, but which 5 percent. The approach taken here will be to give you the 5 percent you need to get going.

The assumption is made that you know how to do the following on an MS-DOS computer:

- Create a new directory called \PIC
- Copy the contents of a floppy disk to the directory \PIC
- Use a simple text editor to create a text file, save it, make a copy of it, print it, and copy it to the directory \PIC

Jameco Part number 136645